A Statistical Model of Skewed Associativity

Pierre Michaud

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Statistical Model of Skewed Associativity

Pierre Michaud

N° 4582

Octobre 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4582--FR+ENG

A Statistical Model of Skewed Associativity

Pierre Michaud

Thème 1 — Réseaux et systèmes
Projet CAPS

Rapport de recherche n° 4582 — Octobre 2002 — 25 pages

Abstract: This paper presents a statistical model of set-associativity, victim caching and skewed-associativity, with an emphasis on skewed-associativity. We show that set-associativity is not efficient when the working-set size is close to the cache size. We refer to this as the *unit working-set* problem. We show that victim-caching is not a practical solution to the unit working-set problem either, although victim caching emulates full associativity for working-sets much larger than the victim buffer itself. On the other hand we show that 2-way skewed associativity emulates full associativity for working-sets up to half the cache size, and that 3-way skewed-associativity is almost equivalent to full associativity, i.e., skewed-associativity is a practical solution to the unit working-set problem.

Key-words: processor, cache, conflicts, skewed-associativity, set-associativity, victim-caching, statistical model

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Un modèle statistique de l'associativité biaisée

Résumé : Cette étude présente un modèle statistique de trois méthodes de résolution des conflits dans les mémoires cache des processeurs : l'associativité par ensembles, le cache de victimes, et l'associativité biaisée. Nous montrons que l'associativité par ensembles n'est pas une méthode efficace lorsque le nombre d'objets à stocker dans le cache est proche de la taille du cache. Cela nécessiterait l'associativité complète, qui est difficile à mettre en oeuvre en pratique. Nous montrons que le cache de victimes permet de simuler l'associativité complète pour un nombre d'objects beaucoup plus grand que le cache de victime lui-même. Cependant, cette méthode ne résout pas le problème. L'associativité biaisée, en revanche, résout pratiquement le problème : elle est proche de l'associativité complète avec seulement trois bancs.

Mots-clés : processeur, cache, conflits, associativité, modèle statistique

1 Introduction

Caching is a basic technique of computer architecture, used for many different purposes. Generally speaking, a cache is a dedicated memory which tries to hold a working-set of data into a limited space. The data may be actual program data, program instructions, address translations, branch predictions, memory dependency predictions, or whatever. Correspondingly, the cache may be a data cache, an instruction cache, a TLB, a BTB, etc... To be general, we will speak in this study of a working-set of *objects*.

Sometimes a cache is not able to hold the whole of a working-set, but only a part of it. In this study, objects that the cache cannot hold are referred to as *missing objects*. Ideally, a cache must minimize the number of missing objects, while providing as fast an access as possible to objects lying in the cache. However these two requirements are antinomic. A fast access time, which implies hardware simplicity, is generally obtained at the cost of some number of missing objects. Missing objects may be due simply to the working-set being larger than the cache. They may also be due to the hashing function used to access the cache, in which case they are called *conflict* misses.

Several solutions have been proposed to remove conflict misses. Set-associativity is the most widely used solution. It consists of splitting the cache into several bank, each bank providing a possible location for an object and being accessed with the same hashing function. We show in this study that, while effective at solving the conflict miss problem when the working-set is much smaller than the cache, set-associativity is not very efficient when the working-set size is close to the cache size. We refer to this as the *unit working-set* problem.

Victim-caching [5] has been proposed in 1990 as a solution to the conflict miss problem in direct mapped caches. It consists of adding a small fully-associative buffer to hold the "victims" of conflicts in the main cache. We show in this study that victim-caching emulates full associativity for working-sets much larger than the victim buffer itself, confirming previously published experimental observations. However, victim-caching is not a practical solution to the unit working-set problem.

Skewed-associativity was introduced in 1993 as an improvement over set-associativity. The basic idea consists of indexing the banks with different hashing functions. Skewed-associativity was initially proposed for instructions and data caches [10]. It was later shown that skewed-associativity, unlike set-associativity, is rather insensitive to spatial locality effects [2], and is efficient for many types of objects [11]. Moreover, several implementable replacement policies were proposed, nearly as good as LRU [11].

This study highlights the fact that the efficiency of skewed-associativity is inherently statistical and does not rely on any characteristics of the applications. We show that 2-way skewed-associativity emulates full-associativity for working-sets up to half the cache size, and that 3-way skewed-associativity is almost equivalent to full associativity. That is, skewed-associativity is a practical solution to the unit working-set problem.

Section 2 describes our approach to the problem and preliminary assumptions for the model. Sections 3, 4 and 5 describe the model for set-associativity, victim-caching and skewed-associativity respectively, with a focus on the unit working-set problem. Section 5 on skewed-associativity constitutes the main part of the study. We emphasize the placement

problem, which is specific to skewed-associativity, and we study three different placement algorithms.

2 Our approach to the problem, and preliminary assumptions

Although motivated by a microarchitecture question, this study makes no assumptions on specific properties of the applications (like spatial and temporal locality), neither on the type of objects. We adopt a static, timeless point of view : given a working-set size, we consider all the possible configurations for keys [1] constituting the working-set as equally likely, and we determine an approximation of the average number of missing objects. We define the *average missing fraction* (**amf**) as the average number of missing objects divided by the working-set size ($amf \in [0..1]$).

We assume the cache is split into w banks, w being the *associativity degree*. Each bank $i \in [1..w]$ is accessed through a hashing functions $\mathcal{H}_i$ which map keys onto cache locations on the bank. If $w = 1$, the cache is said to be *direct-mapped*. Each object in the working-set has w possible locations, one on each bank.

The problem of studying the occupancy of a cache may be viewed as a balls-in-urns (aka balls-in-bins) problem. A ball is associated to a single object in the working-set, but an object may be associated to several balls. An urn generally represents a cache location, or a set of locations, and has room for an unlimited number of balls. We put into each urn the balls corresponding to the objects that **could** be placed in the cache location (or set of locations) associated to the urn. We assume all possible configurations of balls into urns are equiprobable. The mapping of a cache problem onto such balls-in-urns model requires the following assumptions

1. all possible working-sets of n keys in $[0..A-1]$ are equiprobable
2. for every cache location $Y \in [0..N-1]$, the size of the preimage $\mathcal{H}_i^{-1}(Y) = \{X \mid \mathcal{H}_i(X) = Y\}$ equals A/N
3. $A/N \gg n$

The first assumption is part of our argument, as we aim to show that the nature of the efficacy of skewed-associativity is statistical and does not lie in applications characteristics. The second and third assumptions, i.e., that all preimages have the same size and that A/N is much greater than n, is here for the equiprobability of sets of keys to translate into equiprobability of balls-in-urns configurations. Also implicit here is the assumption that $\mathcal{H}_i$ is a good hashing function, i.e., an unbiased one. The third assumption is generally true for most practical cache problems, as the key space is often several orders of magnitude larger than the working-set.

[1] A key is a value in $[0..A-1]$ which uniquely identifies an object, A being the size of the key space

2.1 Classical occupancy problem

In this section, we introduce the formula that will be used as the basis of subsequent statistical models.

We assume N distinguishable urns and n distinguishable balls. A ball is mapped onto a random urn. We assume each urn can hold an unlimited number of balls. There are N^n distinct configurations. We assume all N^n configurations are equiprobable (Maxwell-Boltzmann statistics [3]). We search the average number μ_q of urns holding exactly q balls, with $q \geq 0$ (the case $q = 0$ is sometimes referred to in the literature as the *classical occupancy problem* [3]). We show in Appendix A that, for $N \gg 1$ and $n \gg 1$, μ_q approximates a Poisson distribution:

$$\mu_q \simeq N \frac{(\frac{n}{N})^q}{q!} e^{-\frac{n}{N}} \tag{1}$$

Moreover, for N large enough, the distribution of the number of urns holding q balls is concentrated around the mean, that is, most configurations are close to the average configuration.

2.2 Monte Carlo simulations

Formulas derived analytically in this study are mostly asymptotic approximations. We will use Monte Carlo simulations to validate experimentally these approximations, i.e., show that, under the assumptions of the model, and for realistic cache sizes, the formulas are reasonably accurate. In all our Monte Carlo simulations, we draw 10^4 random configurations, and compute average values from these configurations.

3 Set-associativity

With set-associativity, all the banks are indexed with the same hashing function. Hence the set of possible locations for an object (*set of locations*, for short) is completely determined by its location on a particular bank, say bank 1. Let N be the total number of cache locations and n the number of objects in the working-set. The number of distinct sets of locations is $N_s = N/w$. In the remaining of this study, we define the working-set-to-cache-size ratio

$$\lambda = \frac{n}{N}$$

We can map the set-associativity problem onto a balls-in-urns problem as follows. There is one ball per object and one urn per set of locations. So we have n distinguishable balls and N_s distinguishable urns (we recall that each urn has room for an unlimited number of balls). For urns containing q balls with $q > w$, we count $q - w$ missing objects. Using Formula 1, replacing N with N_s, we obtain the number of unoccupied cache locations

	Monte Carlo					**model**				
λ	w=1	w=2	w=4	w=8	w=16	w=1	w=2	w=4	w=8	w=16
0.10	0.0462	0.0053	0.0001	0.0000	0.0000	0.0484	0.0060	0.0002	0.0000	0.0000
0.20	0.0919	0.0206	0.0018	0.0000	0.0000	0.0937	0.0219	0.0020	0.0000	0.0000
0.30	0.1339	0.0433	0.0071	0.0004	0.0000	0.1361	0.0449	0.0079	0.0005	0.0000
0.40	0.1741	0.0713	0.0183	0.0021	0.0000	0.1758	0.0727	0.0196	0.0025	0.0001
0.50	0.2117	0.1027	0.0359	0.0077	0.0005	0.2131	0.1036	0.0376	0.0084	0.0008
0.60	0.2467	0.1348	0.0599	0.0186	0.0031	0.2480	0.1365	0.0615	0.0203	0.0039
0.70	0.2796	0.1693	0.0887	0.0378	0.0110	0.2808	0.1703	0.0905	0.0397	0.0128
0.80	0.3105	0.2032	0.1212	0.0645	0.0281	0.3117	0.2043	0.1232	0.0668	0.0309
0.90	0.3399	0.2366	0.1568	0.0982	0.0567	0.3406	0.2379	0.1586	0.1007	0.0600
1.00	0.3673	0.2693	0.1938	0.1371	0.0964	0.3679	0.2707	0.1954	0.1396	0.0992
1.10	0.3926	0.3016	0.2309	0.1795	0.1426	0.3935	0.3024	0.2325	0.1815	0.1456
1.20	0.4170	0.3320	0.2680	0.2230	0.1935	0.4177	0.3330	0.2693	0.2246	0.1953
1.30	0.4396	0.3616	0.3040	0.2658	0.2437	0.4404	0.3622	0.3051	0.2672	0.2451
1.40	0.4614	0.3893	0.3387	0.3073	0.2915	0.4619	0.3900	0.3395	0.3083	0.2925
1.50	0.4816	0.4156	0.3713	0.3465	0.3360	0.4821	0.4163	0.3722	0.3472	0.3364

Table 1: Set-associativity : $amf(w, \lambda)$ obtained with Monte Carlo simulations on 240 cache locations (left part) and amf obtained with Formulas 2 and 3 (right part)

$$\sum_{q=0}^{w-1} N_s \frac{(\lambda w)^q}{q!} e^{-\lambda w}(w-q)$$

Note that this approximation is meaningful only for $N_s \gg 1$, that is, $w \ll N$. We define the *average unoccupied fraction* (**auf**) as the number of unoccupied cache locations divided by the total number of cache locations N. For a set-associative cache, the auf is

$$auf(w,\lambda) \simeq e^{-\lambda w} \times \sum_{q=0}^{w-1} \frac{(\lambda w)^q}{q!}(1-\frac{q}{w}) \tag{2}$$

The amf can be obtained from the auf

$$amf(w,\lambda) = 1 - \frac{1-auf(w,\lambda)}{\lambda} \tag{3}$$

Tables 1 shows the amf obtained with Formulas 2 and 3 for various values of λ and w. It also shows the amf obtained with Monte Carlo simulations on 240 cache locations. [2] It can be observed that Formula 2 provides a reasonably accurate approximation. [3]

We can distinguish three interesting regions for λ values : $\lambda > 2$, $\lambda \ll 1$, and λ close to 1. For $\lambda > 2$, the amf of a direct-mapped cache is close to that of a fully-associative one, which can be obtained from Formula 3

$$amf(\infty,\lambda) = \begin{cases} 0 & \text{for } \lambda \leq 1 \\ 1-\frac{1}{\lambda} & \text{for } \lambda > 1 \end{cases}$$

In other words, the benefit of increasing the associativity diminishes quickly as we increase the working-set size beyond $\lambda = 1$. For $\lambda \ll 1$, the amf can be approximated as

$$amf(w,\lambda) \approx \frac{(\lambda w)^w}{(w+1)!} \tag{4}$$

i.e., the amf behaves like a polynomial in λ of degree w. In particular, $amf(1,\lambda)$ has a first derivative which is positive at $\lambda = 0$. This weakness of direct-mapped caches is well known, though not intuitive, and is closely related to the so-called birthday paradox.

[2] We chose 240 only for practical reasons, as it is a multiple of 16, 10, and 3 (we will simulate 3-way associativity later on).

[3] The approximation becomes better with more cache locations.

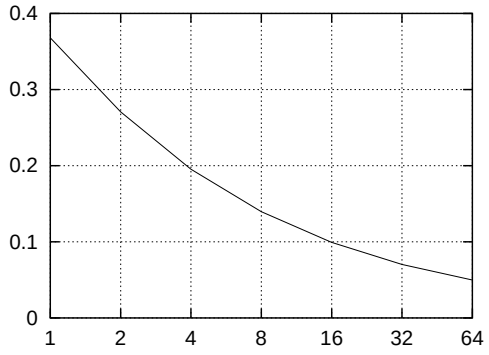

Figure 1: Set-associativity : amf as a function of w for $\lambda = 1$ (Formulas 2 and 3)

The unit working-set problem. In general, two-way set-associativity is sufficient for most problems involving a working-set much smaller than the cache. However, results on Table 1 show that when the working-set is of the same order of magnitude as the cache size, $amf(2, \lambda)$ is significant. The worst case is $\lambda = 1$.

Figure 1 shows the curve of the amf as a function of w for $\lambda = 1$, obtained with Formulas 2 and 3. The amf decreases slowly as w increases (as long as $N \gg w$). Even with $w = 64$, the amf is still significant, around 5%. Set-associativity is not an efficient solution for removing conflicts when the working-set size is close to the cache size. We refer to this problem as the *unit working-set* problem.

What the model does not explain lies in application characteristics. Some empirical knowledge about caches is not explained by the model. For example, the well-known rule of thumb, which states that a 2-way set-associative instruction/data cache is equivalent to a direct-mapped cache twice larger, is not confirmed by the model. This shows that such "rule" depends on application characteristics (probably spatial locality in this case), and should not be taken as a property of set-associativity in general.

4 Victim caching

A victim buffer is a small fully-associative cache, alongside the main cache, which purpose is to hold the objects that are "victim" of conflicts in the main cache. Victim caching was proposed by Norman Jouppi [5] as an alternative to set-associativity. It is possible to generalize victim-caching to a set-associative main cache. However, victim-caching makes sense only for a low-associativity main cache and a victim buffer much smaller than the main cache.

Parameter λ still denotes the ratio of the working-set size n over the main cache size N. We introduce the ratio v of the victim-buffer size N_v over the main cache,

$$v = \frac{N_v}{N}$$

Missing objects are the objects that neither the main cache nor the victim buffer can hold. The corresponding average missing fraction will be denoted amf_v. The notation $amf(w, \lambda)$ refers to the fraction of objects that the main cache is not able to hold. It is obtained with Formulas 2 and 3. Assuming most balls-in-urns configurations are close to the average configuration, the amf_v can be approximated as

$$amf_v(w, v, \lambda) \approx max(0, amf(w, \lambda) - \frac{v}{\lambda}) \tag{5}$$

The interesting parameter for victim-caching is the value $\lambda_0(v)$ below which the afm_v is null. This can be determined as the solution of the equation

$$amf(w, \lambda) - \frac{v}{\lambda} = 0$$

For a direct-mapped cache, we get a simple approximation using Formula 4 with $w = 1$

$$\lambda_0(v) \approx \sqrt{2v}$$

For instance, with $v = 0.01$, we get $\lambda_0 \approx 0.14$. In other words, on this example, victim-caching emulates full associativity for working-sets up to 14 times the victim buffer size. However, for $\lambda > \lambda_0$, the afm_v increases very fast with λ.

Victim-caching is very efficient for removing "birthday paradox" misses encountered with direct-mapping. However, victim-caching is not a practical solution to the unit working-set problem. Figure 2 shows the amf_v as a function of v for $\lambda = 1$ and for a direct-mapped, 2-way and 4-way set-associative main cache. It requires a 4-way set-associative main cache and a victim buffer 1/5 the main cache size for the amf_v to become insignificant. Such configuration is no longer in the spirit of victim-caching.

5 Skewed-associativity

Skewed-associativity was introduced by André Seznec [10, 2, 11] as an improvement over set-associativity. A w-way skewed-associative cache consists of w tables of N/w entries each. As in the case of set-associativity, an object has w possible cache locations, one on each bank. However, unlike set-associativity, each bank $i \in [1..w]$ in a skewed-associative cache is indexed with a hashing function $\mathcal{H}_i$ different from that of the other banks. A major consequence of this skewing of the indexing functions is that the global cache occupancy depends on which of the w possible locations is chosen for an object. We refer to this as the *placement* problem.

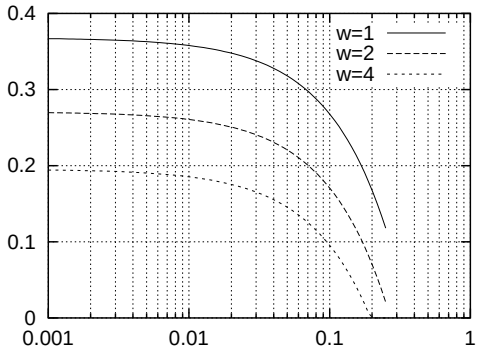

Figure 2: Curves of the amf_v as a function of v for $\lambda = 1$

5.1 The placement problem on skewed-associative caches

The placement problem was irrelevant for set-associativity. [4] For skewed-associative caches, it is a crucial problem. In this study, we will consider three placement methods: ordered-banks placement (**OBP**), quasi-optimal placement (**QOP**), and iterative random placement (**IRP**). The first placement algorithm provides an example of one-pass method. The second placement algorithm, on the other hand, provides a practical lower bound for the amf of a skewed-associative cache. The third placement algorithm corresponds to the practical case [2], and converges toward an optimal placement.

5.2 Skewed-associativity as a balls-in-urns problem

We can map the skewed-associativity problem onto a balls-in-urns problem as follows. There is one urn per cache location (each urn has room for an unlimited number of balls). The set U of all urns is partitioned into w disjoint subsets of N/w urns, each subset corresponding to a different bank. A working-set of n objects is associated to w balls-in-urns configurations, one configuration per bank. Each object is represented by w balls, i.e., one ball per bank configuration. We are studying the global configuration resulting from the combination of these w per-bank configurations. The total number of global configurations is

$$\left(\frac{N}{w}\right)^{wn}$$

[4] The placement problem is distinct from the usual replacement problem in caches. The usual replacement problems deals with the fact that some objects are accessed more frequently than others. When one among several objects must be evicted from the cache, a good non-oracle replacement policy, like LRU, tries to identify the object referenced the least frequently. With our static point of view, all objects are "equal", and there is no replacement question.

	Monte Carlo			model		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.10	0.0007	0.0000	0.0000	0.0009	0.0000	0.0000
0.20	0.0056	0.0001	0.0000	0.0060	0.0001	0.0000
0.30	0.0169	0.0009	0.0000	0.0176	0.0010	0.0000
0.40	0.0347	0.0042	0.0002	0.0358	0.0046	0.0003
0.50	0.0588	0.0129	0.0017	0.0601	0.0135	0.0020
0.60	0.0877	0.0289	0.0076	0.0892	0.0298	0.0082
0.70	0.1205	0.0527	0.0214	0.1217	0.0539	0.0226
0.80	0.1550	0.0842	0.0461	0.1565	0.0853	0.0475
0.90	0.1907	0.1205	0.0810	0.1923	0.1220	0.0823
1.00	0.2274	0.1607	0.1229	0.2283	0.1621	0.1242
1.10	0.2630	0.2025	0.1687	0.2638	0.2036	0.1700
1.20	0.2977	0.2441	0.2158	0.2983	0.2451	0.2168
1.30	0.3309	0.2846	0.2617	0.3314	0.2854	0.2624
1.40	0.3621	0.3232	0.3052	0.3630	0.3238	0.3057
1.50	0.3922	0.3594	0.3456	0.3928	0.3599	0.3460

Table 2: Skewed-associativity : amf_{obp} obtained with Monte Carlo simulations on 240 cache locations (left part) and amf_{obp} obtained with Formula 6 (right part)

We assume all these global configurations are equiprobable. In addition to the conditions listed in Section 2 for each hashing function $\mathcal{H}_i$ separately, this assumption of equiprobability requires that for every w-uple of cache locations $(Y_1, ..., Y_w)$,

$$card\Big(\bigcap_{i=1}^{w} \mathcal{H}_i^{-1}(Y_i)\Big) = \frac{A}{(N/w)^w} \gg n$$

with A the size of the key space. For short, we will say that the hashing functions $\mathcal{H}_i$ are "orthogonal". Orthogonality is more restrictive than the *inter-bank dispersion* property as defined in [2] (namely, that $card(\mathcal{H}_i^{-1}(Y_i) \cap \mathcal{H}_j^{-1}(Y_j)) = \frac{A}{(N/w)^2}$ for $i \neq j$). However, orthogonality implies inter-bank dispersion. With this assumption of orthogonality, the w per-bank balls-in-urns configurations have independent statistics.

5.3 Ordered-banks placement (OBP)

The OBP algorithm is a one-pass algorithm. We assume that the banks are numbered from 1 to w, and the objects are numbered from 1 to n. The OBP algorithm tries to place objects $k = 1...n$ sequentially. If the location for object k on bank 1 is empty, we place object k on bank 1, else we look at bank 2. If the location for object k on bank 2 is empty, we place object k on bank 2, else we look at bank 3, and so on ... If we were not able to place object k

because all w locations were already occupied, we count object k in the missing objects, and we repeat the process for object $k+1$. Note that with this algorithm, banc i will receive, on average, more objects than bank $j > i$.

We use notation n_i for the total number of objects placed on banks 1 to i, and notation $f_i = \frac{n_i}{n}$ for the corresponding fraction. Using formula 1 with $q = 0$ and replacing N with N/w, the average fraction of objects placed on bank 1 can be approximated as

$$f_1 \simeq \frac{1 - e^{-\lambda w}}{\lambda w}$$

Assuming most configurations are close to the average configuration, we obtain f_i recursively

$$f_i \simeq f_{i-1} + \frac{1 - e^{-(1-f_{i-1})\lambda w}}{\lambda w}$$

Finally, the global amf_{obp} is

$$amf_{obp} = 1 - f_w \tag{6}$$

Table 2 shows the amf_{obp} obtained with Monte Carlo simulations on 240 cache locations and that obtained with Formula 6. We observe that Formula 6 is a reasonably accurate approximation of the ordered-banks placement behavior. Comparing the numbers in Tables 1 and 2, we see that a skewed-associative cache with OBP has a lower amf than a set-associative cache with the same associativity. The behavior of a 2-way skewed-associative cache with OBP is between that of a 2-way and 4-way set-associative cache. This result shows that skewed-associativity is efficient even with a "poor" placement method. Nevertheless, OBP does not really solve the unit working-set problem.

5.4 Quasi-optimal placement (QOP)

The QOP algorithm places the objects one at a time, but with a global knowledge of all the objects. We call a *free* object an object that is not yet placed at a given "time" of the algorithm. Before the placement algorithm starts, all n objects are free. We denote $F(u)$ the set of **free** objects that have balls in urn u. We denote $V_t \subset U$ the subset of urns corresponding to empty locations at time t. Initially, $V_0 = U$. The QOP algorithm is defined recursively as follows:

1. If there exists any urn $u \in V_t$ such that $card(F(u)) = 1$, select one such u and place the object from $F(u)$ in the location associated to u. After that, $V_{t+1} = V_t \setminus \{u\}$.
2. **Otherwise**, if there exists any urn $u \in V_t$ such that $card(F(u)) > 1$, select one such u, select an object from $F(u)$, and place it in the location associated to u. After that, $V_{t+1} = V_t \setminus \{u\}$.

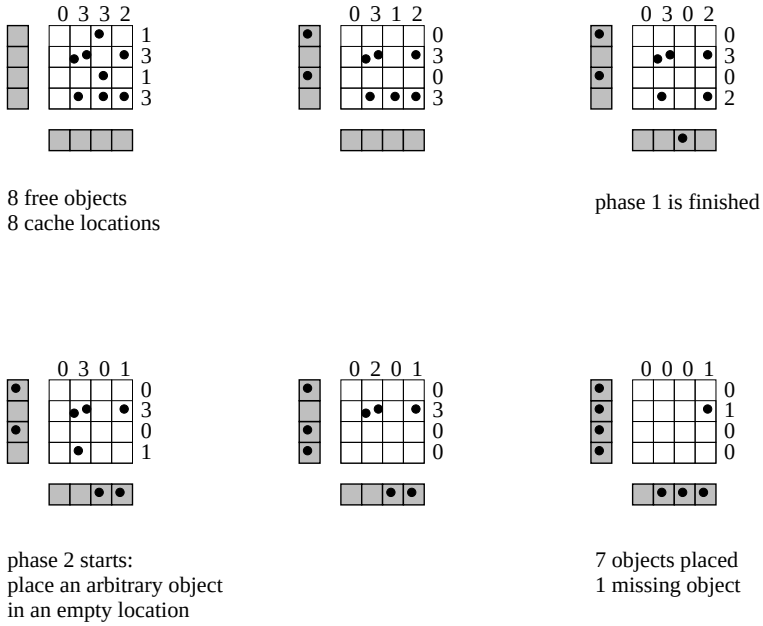

Figure 3: Example of QOP placement on a 2-way skewed-associative cache. A dot represents an object in the working-set. Its coordinates on the array represent its locations on the two banks. There are 4 locations on each bank, for a total of 8 locations, each location being associated with an urn. Numbers on the upper side and rightmost side represent $card(F(u))$ for the associated urn.

When we have several possibilities for choosing an urn or an object, we choose according to a predefined order. The algorithm ends when it is no longer possible to place new objects in this way. Free objects remaining at that time are missing objects. Figure 3 shows how the QOP algorithm works on an example.

Table 3 shows the amf_{qop} obtained with Monte Carlo simulations on 240 cache locations. If we focus on row $\lambda = 0.9$, we see that the unit working-set problem is practically solved for an associativity degree of 3.

5.4.1 The QOP algorithm is optimal in the case $w = 2$

It can be demonstrated that the QOP algorithm is optimal in the case $w = 2$. We distinguish two phases of the algorithm (we found the algorithm easier to analyze this way). Phase 1 of the algorithm ends the first time all urns $u \in V_t$ are such that $card(F(u)) \neq 1$. After that time, and until the algorithm ends, this is phase 2. It should be noted that the objects that are placed at the end of phase 1 do not depend on the way we selected the urns. They

	Monte Carlo			**model**
λ	w=2	w=3	w=4	w=2
0.10	0.0000	0.0000	0.0000	0.0000
0.20	0.0000	0.0000	0.0000	0.0000
0.30	0.0000	0.0000	0.0000	0.0000
0.40	0.0002	0.0000	0.0000	0.0000
0.50	0.0012	0.0000	0.0000	0.0000
0.60	0.0077	0.0000	0.0000	0.0062
0.70	0.0305	0.0000	0.0000	0.0309
0.80	0.0679	0.0000	0.0000	0.0693
0.90	0.1130	0.0041	0.0000	0.1146
1.00	0.1607	0.0607	0.0237	0.1619
1.10	0.2077	0.1293	0.1030	0.2085
1.20	0.2524	0.1917	0.1737	0.2531
1.30	0.2945	0.2474	0.2350	0.2949
1.40	0.3330	0.2969	0.2883	0.3337
1.50	0.3691	0.3409	0.3349	0.3695

Table 3: Skewed-associativity : amf_{qop} obtained with Monte Carlo simulations on 240 cache locations (left part) and amf_{qop} for $w = 2$ obtained with Formula 12 (right part)

depend solely on the initial configuration. The placements done during phase 1 are optimal because the objects placed occupy locations that could not be occupied by any other object. So the question of the optimality of the QOP algorithm arises when entering phase 2.

We denote t_K the time when phase 2 starts. We denote K the set of free objects at time t_K, and U_K the set of urns containing any ball associated to objects in K (on the example of Figure 3, K consists of 5 objects and U_K consists of 4 urns).

We show in this section that the QOP algorithm is optimal in the case $w = 2$. At time t_K, there are $card(K)$ free objects, and $card(U_K)$ possible locations for these objects. It should be noted that $U_K \subset V_{t_K}$. Moreover, $card(F(u)) \geq 2$ for all $u \in U_K$. The maximum number of objects that can be placed during phase 2 is

$$min(card(K), card(U_K))$$

In the case $w = 2$, we have $card(K) \geq card(U_K)$. This can be seen by summing up the number of balls in U_K, and dividing by w because each object in $F(U_K)$ (before phase 2 starts) is associated to w balls

$$\begin{aligned} card(K) &= \frac{1}{w} \sum_{u \in U_K} card(F(u)) \\ &\geq \frac{2}{w} card(U_K) \\ &= card(U_K) \end{aligned}$$

Consequently, in the case $w = 2$, the maximum number of objects that can be placed during phase 2 is $card(U_K)$. Actually, it is possible to place an object in each of the $card(U_K)$ locations. To see this, we show that, at any time t,

$$\begin{cases} card(\{u \in V_t \cap U_K \mid card(F(u)) = 0\}) = 0 \\ card(\{u \in V_t \cap U_K \mid card(F(u)) = 1\}) \leq 1 \end{cases} \quad (7)$$

If this assertion holds for $t \geq t_K$, $V_t \cap U_K$ decreases as long as $V_t \cap U_K \neq \emptyset$, and we are able to fill all the locations associated to urns in U_K. We show assertion 7 inductively. It is obviously true at time t_K. We assume assertion 7 is true at time t, and we show it remains true at time $t + 1$ after placing an object. There are two cases :

- For all urns $u \in V_t \cap U_K$, $card(F(u)) \geq 2$. After placing the object, $card(F(u))$ is decremented for all w urns u holding a ball associated to the object. One of these w urns is not in V_{t+1}. The $w - 1$ other urns may have $card(F(u)) = 1$. However, as $w = 2$, there is at most one such urn, hence assertion 7 remains true.
- There exists an urn $v \in V_t \cap U_K$ such that $card(F(v)) = 1$. From assertion 7, $card(F(u)) \geq 2$ for $u \in V_t \cap U_K$ different from v. The object from $F(v)$ is placed

in the location associated to v. After placing the object, $card(F(v)) = 0$. As $v \notin V_{t+1}$, the first part of the assertion remains true. We may have $card(F(u)) = 1$ for the $w-1$ other urns holding balls associated with the object, but as $w = 2$ there is at most one such urn, hence assertion 7 remains true.

5.4.2 Approximation of amf_{qop} in the case $w = 2$

To derive an approximation of amf_{qop} in the case $w = 2$, we need to derive an approximation of the average value of $card(K)$ and $card(U_K)$ respectively. As previously, we will assume that most configurations are close to the average configuration.

We recall that set K is precisely defined for a given configuration, as the precise set of objects that are placed at the end of phase 1 does not depend on how we select the urns. On the other hand, the precise sets of objects that are placed on a given bank may depend on the way urns are selected. Nevertheless, for a given configuration, and for a given bank i, there exists a definite set S_i of objects that cannot be placed on bank i during phase 1, however urns are selected. Objects in K are objects that could not be placed on any bank during phase 1 :

$$K = \bigcap_{i=1}^{w} S_i$$

Let β be the probability that an object belong to S_i. The problem is symmetric (the banks have the same size), so β is the same for all the banks. We have

$$card(K) = n\beta^w \tag{8}$$

It is also possible to derive $card(K)$ by going back to the algorithm. Let us consider the set J of objects defined as

$$J = \bigcap_{i=1}^{w-1} S_i$$

i.e., J is the union of K and the set of objects than can be placed only on bank w during phase 1. The number of objects in J is

$$card(J) = n\beta^{w-1}$$

This is the number of free objects remaining when we place all the objects that can be placed during phase 1 except the objects that can be placed only on bank w. The number of objects that can be placed only on bank w equals the number of urns u on bank w such that $card(F(u)) = 1$. This number can be obtained with Formula 1 for $q = 1$, replacing n with $card(J)$ and N with N/w

$$card(J) - card(K) \simeq n\beta^{w-1}e^{-\lambda w\beta^{w-1}}$$

We get a second expression for $card(K)$:

$$card(K) \simeq n\beta^{w-1}\left(1 - e^{-\lambda w \beta^{w-1}}\right) \tag{9}$$

By equating expressions 8 and 9, we obtain an equation for β

$$\beta + e^{-\lambda w \beta^{w-1}} = 1 \tag{10}$$

One trivial solution to this equation is $\beta = 0$. For $w = 2$, this is the only solution in [0..1] when $\lambda \leq 0.5$ It means that 2-way skewed-associativity with QOP emulates full associativity for $\lambda \leq 0.5$.

When there are several solutions to Equation 10 in [0..1], we will take the one closest to 1. [5] Appendix B compares $card(K)/n$ obtained with Monte Carlo simulations and that obtained with Equation 10 and Formula 8.

Now we seek $card(U_K)$. Like K, U_K depends solely on the initial configuration. From the symmetry of the problem, urns in U_K are spread equally on the w banks on average. Let us place all the objects that are not in J, as we did previously. There remains $n\beta^{w-1}$ free objects. Urns on bank w that belong to U_K are those urns u such that $card(F(u)) > 1$. The number of such urns can be obtained with Formula 1 with $q = 0$ and $q = 1$, replacing n with $n\beta^{w-1}$ and N with N/w

$$\frac{card(U_K)}{w} \simeq \frac{N}{w} - \frac{N}{w}e^{-\lambda w \beta^{w-1}} - n\beta^{w-1}e^{-\lambda w \beta^{w-1}}$$

This simplifies to

$$card(U_K) \simeq N\beta - nw(1-\beta)\beta^{w-1} \tag{11}$$

In the case $w = 2$, we have

$$amf_{qop} = \frac{card(K) - card(U_K)}{n}$$

Using expressions 8 and 11, we obtain

$$amf_{qop}(2,\lambda) \simeq 2\beta - \beta^2 - \frac{\beta}{\lambda} \tag{12}$$

Table 3 shows $amf_{qop}(2,\lambda)$ obtained with Formula 12.

[5] We were not able to find a justification for this.

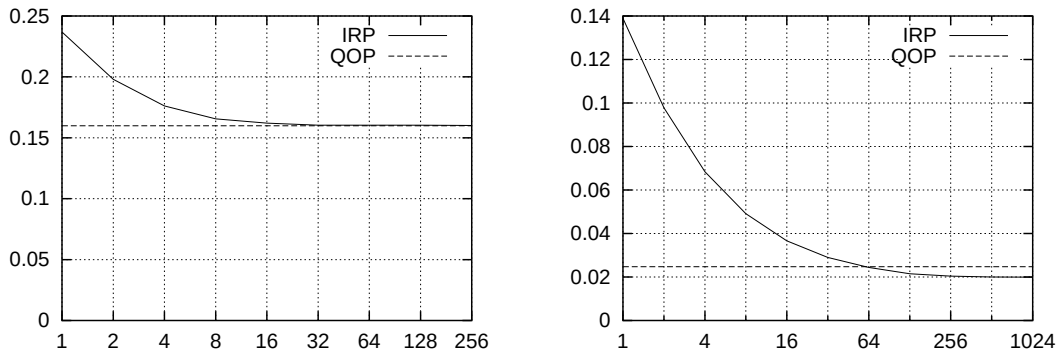

Figure 4: Skewed-associativity: amf_{irp} for $\lambda = 1$ as a function of the number of passes (Monte Carlo simulations on 160 locations). The left graph is for $w = 2$, the right graph is for $w = 4$.

5.4.3 The QOP algorithm is not optimal in the case $w > 2$

During phase 2 of the QOP algorithm, we scan the set of objects according to a predefined order, and for each object, we try the banks also according to a predefined order, until we are able to place an object. In our Monte Carlo simulations, we experimented two different ways to order the objects and the banks. For $w > 2$, on some configurations, the number of missing objects was different with the two methods, although on average, as expected, there was no significant difference. This shows that the QOP algorithm is not optimal in a general way.

5.5 Iterative random placement (IRP)

We have seen that carefully placing the objects may significantly lower the amf. However, the QOP algorithm requires some global knowledge (i.e., $card(F(u))$) that is typically difficult to obtain in real microarchitecture situations. Nevertheless, it is possible to approximate QOP by using a multi-pass algorithm. The IRP algorithm works as follows. The objects are numbered from 1 to n, and we scan the set of objects repeatedly in that order, going back to object 1 after object n. If the object is already placed, we do nothing, and simply go to the next object. Otherwise, if the object is not placed, there are two cases :

- If there exists an empty location among the set of locations, we place the object in one such location. If there are several empty locations, we choose one randomly.
- Otherwise, if there is no empty location, we choose a random bank, we evict the object already stored on that bank, and we place the new object.

	Monte Carlo			**model**		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.30	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.40	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000
0.50	0.0013	0.0000	0.0000	0.0000	0.0000	0.0000
0.60	0.0079	0.0000	0.0000	0.0062	0.0000	0.0000
0.70	0.0302	0.0000	0.0000	0.0309	0.0000	0.0000
0.80	0.0679	0.0000	0.0000	0.0693	0.0000	0.0000
0.90	0.1132	0.0017	0.0000	0.1146	0.0000	0.0000
1.00	0.1609	0.0597	0.0204	0.1619	0.0608	0.0209
1.10	0.2074	0.1291	0.1027	0.2085	0.1299	0.1032
1.20	0.2521	0.1917	0.1737	0.2531	0.1922	0.1740
1.30	0.2944	0.2474	0.2350	0.2949	0.2478	0.2352
1.40	0.3332	0.2969	0.2884	0.3337	0.2972	0.2884
1.50	0.3691	0.3410	0.3350	0.3695	0.3412	0.3350

Table 4: Skewed-associativity : amf_{irp} obtained with Monte Carlo simulations on 240 cache locations after 1000 passes (left part) and amf_{min} obtained with Formula 13 (right part)

The algorithm stops after an arbitrary number of passes. Missing objects are the objects that are out of the cache when the algorithm stops. The IRP algorithm was experimented in [2]. It is typically what happens in a real cache problem.

It should be noted that the number of objects placed cannot decrease. The first time we place an object, the placement may not be optimal. However, after several passes, we will find an optimal placement for the object, e.g., a location that can be occupied by no other object (in that respect, randomly choosing the bank is essential because it permits trying all the banks). Eventually, after several passes, the cache occupancy converges toward an optimal placement. This was referred to as "self data reorganization" in [2].

Table 4 shows the amf_{irp} obtained with Monte Carlo simulations on 240 cache locations after 1000 passes, which practically provides an optimal placement. Comparison with Table 3 shows that, although not optimal in the case $w > 2$, the QOP algorithm is not far from optimality. Actually, it is possible to approximate the amf of an optimal placement by assuming that an optimal second phase of the QOP algorithm could place $min(card(K), card(U_K))$ objects. Although not true for all configurations, [6] this is true for a majority of configurations. We obtain the following formula :

[6] We were able to find a counterexample for $w = 3$.

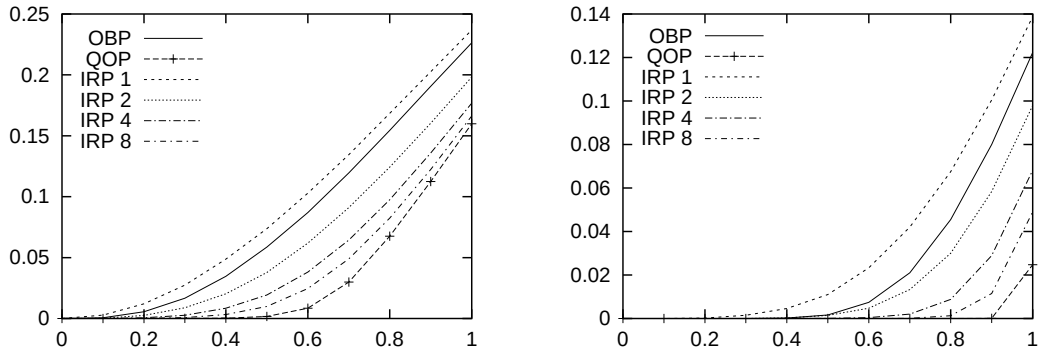

Figure 5: Skewed-associativity: amf_{qop}, amf_{obp} and amf_{irp} after 1, 2, 4, and 8 passes (λ on the x-axis). The left graph is for $w = 2$ and the right graph is for $w = 4$.

$$amf_{min} \doteq max\Big(0, \frac{card(K) - card(U_K)}{n}\Big)$$

and as a function of λ and w

$$amf_{min}(w, \lambda) = max(0, \beta^w + w(1-\beta)\beta^{w-1} - \frac{\beta}{\lambda}) \quad (13)$$

Results obtained with Formula 13 are presented on Table 4, taking for β the solution of Equation 10 closest to 1. This matches the results of Monte Carlo simulations reasonably well.

It was shown experimentally in [11] that 2-way skewed-associativity is generally better than 4-way set-associativity and 4-way skewed-associativity is generally better than 16-way set-associativity. These observations are coherent with results on Tables 1 and 4, whatever the working-set size.

Dynamic behavior of IRP. Figure 4 shows the amf_{irp} for $\lambda = 1$ as a function of the number of passes, for $w = 2$ and $w = 4$. We observe that convergence is more "difficult" for $w = 4$ ($\sim$ 100 passes) than for $w = 2$ ($\sim$ 10 passes). An intuitive explanation is that, with more possible locations for an object, it takes longer to find the "right" location.

Figure 5 compares amf_{qop}, amf_{obp}, and amf_{irp} after 1, 2, 4, and 8 passes, for $w = 2$ and $w = 4$. It can be seen that only a few passes are necessary until amf_{irp} comes close to amf_{qop}.

One-pass placement: OBP is better than random. On Figure 5, it is interesting to note that, after one pass, IRP is not as good as OBP: on the first pass, it is better trying the banks according to a predefined order rather than choosing a random bank.

This can be understood intuitively as follows. Let us consider a 2-way skewed associative cache with N locations, and m objects already placed. We seek to place one extra object. Let us assume that, among these m objects, xm have been placed on one bank, and $(1-x)m$ have been placed on the other bank. The probability that the new object will conflict with objects already placed is

$$\frac{xm}{N/2} \times \frac{(1-x)m}{N/2} = x(1-x)\left(\frac{2m}{N}\right)^2$$

This probability is maximum for $x = 1/2$, i.e., when both banks hold the same number of objects. A random placement will place statistically the same number of objects on both banks. Random placement is a pessimal strategy for a one-pass placement. The OBP algorithm, on the other hand, will put more objects on one bank than on the other bank. A similar reasoning holds for associativity degrees greater than 2.

6 Related works

Most previously published cache models focus on set-associativity (e.g., [1, 8]), as it is the most widely used technique. The goal of these models is, in general, to predict the dynamic miss ratio. In this kind of effort, the main difficulty consists of modeling application characteristics, like spatial and temporal locality, and their effects. In some of these works [6, 8], the fraction of static conflicts was derived with approaches similar to ours.

A few previous works have studied the performance of skewed-associativity [10, 2, 11, 12]. These are mostly experimental studies, that also treat specific implementation problems, like defining adequate hashing functions, or emulating LRU replacements.

To our knowledge, the work closest to ours is [7]. It derives amf_{obp} for $w = 2$ and $\lambda = 1$. However, the essential question of the optimal placement is not addressed in this work.

7 Conclusion

With a combination of Monte Carlo simulations and analytical modeling, we have shown that set-associativity is not an efficient method to remove conflict misses when the working-set size is close to the cache size, which we call the unit working-set problem. We have shown that victim-caching, although able to emulate full associativity for working-sets much larger than the victim buffer itself, is not a practical solution to the unit working-set problem either.

On the other hand, we have shown that 2-way skewed-associativity emulates full associativity for working-sets up to half the cache size, and that 3-way skewed-associativity

emulates full associativity for working-sets up to 90% the cache size, i.e., it is almost equivalent to full associativity. Moreover, we have shown that the efficiency of skewed-associativity is inherently statistical and does not depend on any characteristics of the applications.

From this study, we conclude that the hardware complexity of any proposed solution for providing full associativity should not exceed that of 4-way skewed-associativity, or requires solid arguments against it.

A Balls-in-urns configurations

We are using in this appendix standard methods of combinatorial enumeration. We refer to [9, 4], or any other book on the topic, for an introduction to these methods. We are studying here the configurations of n distinguishable balls into N distinguishable urns. We model an urn as a labeled structure, as described in [4]. The exponential generating function (EGF) for a single urn is

$$u(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!} = e^z$$

The EGF for N urns is

$$U(z) = u(z)^N = e^{zN}$$

To obtain the number of configurations of n balls into N urns, we extract the coefficient of z^n in the Taylor expansion of $U(z)$ and multiply it by $n!$

$$n![z^n]U(z) = N^n$$

which is the expected result. Until now, we have done nothing but decomposing the configurations in a way allowing to extract some information. In particular, we are interested in knowing the number of configurations with a fixed number k of urns containing exactly q balls. To this aim, we "mark" the urn containing q balls in the EGF $u(z)$ using an extra variable x. We obtain the following bivariate generating function (BGF) for a single urn

$$u(z,x) = e^z + (x-1)\frac{z^q}{q!}$$

The BGF for a sequence of N distinguishable urns is

$$U(z,x) = u(z,x)^N$$

The number of configurations with (exactly) k urns containing (exactly) q balls is obtained as

$$c_k = n![z^n x^k]U(z,x)$$

However, instead of computing explicitly the distribution, we can extract the average $\overline{k}$ directly from the ordinary generating function (OGF) of c_k

$$U_n(x) = n![z^n]U(z,x) = \sum_{k=0}^{\infty} c_k x^k$$

The average number μ_q of urns with q balls can be obtained as

$$\mu_q = \overline{k} = \frac{U_n^{'}(1)}{U_n(1)} = \frac{n![z^n]\frac{z^q}{q!}Ne^{z(N-1)}}{N^n} = N\binom{n}{q}\frac{(N-1)^{n-q}}{N^n}$$

When $N \gg 1$ and $n \gg 1$, the average number μ_q of urns with q balls can be approximated using Stirling's approximation

$$\mu_q \simeq N\frac{(\frac{n}{N})^q}{q!}e^{-\frac{n}{N}}$$

In other words, the distribution of urn sizes converges toward a Poisson distribution. The moment of order 2 can be obtained as

$$\overline{k^2} = \frac{U_n^{''}(1) + U_n^{'}(1)}{U_n(1)}$$

It can be shown that the standard deviation $\sigma = \sqrt{\overline{k^2} - \overline{k}^2}$ becomes negligible compared to μ_q as we increase the number of urns. This concentration of the distribution around the mean is called *convergence in probability* [4]. Practically, it means that most configurations are close to the average configuration.

B Skewed-associativity with QOP placement : $card(K)/n$

We are looking here at the fraction of free objects remaining at the end of phase 1 of the QOP algorithm, namely, $card(K)/n$. Table 5 shows $card(K)/n$ obtained with Monte Carlo simulations on 240 cache locations and that obtained with Formula 8, taking for β the greatest solution of Equation 10 in [0..1].

It can be observed that the model matches the actual behavior reasonably well, although the rise of the curve is a little steeper with the model (as expected, we observed that the model matches the actual behavior better with more cache locations, as we get closer to the asymptotic behavior).

It should be noted that there is a threshold λ_0 below which $card(K)$ is null, or very small. The increase of $card(K)$ for $\lambda > \lambda_0$ is steep, especially for $w > 2$. For instance, for $w = 4$, K is almost empty at $\lambda = 0.7$, and it rises to almost 70% of the working-set at $\lambda = 0.8$. This threshold effect is rather counterintuitive.

	Monte Carlo			**model**		
λ	w=2	w=3	w=4	w=2	w=3	w=4
0.10	0.0017	0.0000	0.0000	0.0000	0.0000	0.0000
0.20	0.0039	0.0001	0.0000	0.0000	0.0000	0.0000
0.30	0.0072	0.0002	0.0000	0.0000	0.0000	0.0000
0.40	0.0144	0.0002	0.0000	0.0000	0.0000	0.0000
0.50	0.0349	0.0003	0.0000	0.0000	0.0000	0.0000
0.60	0.1070	0.0004	0.0000	0.0984	0.0000	0.0000
0.70	0.2559	0.0020	0.0121	0.2611	0.0000	0.0000
0.80	0.4085	0.2445	0.6790	0.4121	0.0000	0.6890
0.90	0.5342	0.6613	0.8435	0.5365	0.6611	0.8410
1.00	0.6343	0.7849	0.9076	0.6349	0.7835	0.9061
1.10	0.7117	0.8551	0.9428	0.7119	0.8539	0.9417
1.20	0.7722	0.8997	0.9636	0.7719	0.8986	0.9629
1.30	0.8195	0.9291	0.9765	0.8188	0.9285	0.9760
1.40	0.8560	0.9496	0.9849	0.8556	0.9489	0.9843
1.50	0.8853	0.9638	0.9900	0.8845	0.9632	0.9897

Table 5: Skewed-associativity : $card(K)/n$ obtained with Monte Carlo simulations on 240 cache locations (left part) and $card(K)/n$ obtained with Formula 8 (right part)

References

[1] A. Agarwal, M. Horowitz, and J. Hennessy. An analytical cache model. *ACM Transactions on Computer Systems*, 7(2):184–215, May 1989.

[2] F. Bodin and A. Seznec. Skewed associativity improves program performance and enhances predictability. *IEEE Transactions on Computers*, 46(5):530–544, May 1997.

[3] W. Feller. *An introduction to probability theory and its applications*, volume 1. Wiley, second edition, 1957.

[4] P. Flajolet and R. Sedgewick. Analytic combinatorics - Symbolic combinatorics. http://algo.inria.fr/flajolet/Publications, May 2002.

[5] N. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, 1990.

[6] R.E. Kessler and M.D. Hill. Page placement algorithms for large real-indexed caches. *ACM Transactions on Computer Systems*, 10(4):338–359, November 1992.

[7] S.N. Majumdar and J. Radhakrishnan. Analytical studies of strategies for utilization of cache memory in computers. arXiv:cond-mat/0001090, January 2000. http://arxiv.org/abs/cond-mat/0001090.

[8] M. Schlansker and R. Shaw. Randomization and associativity in the design of placement-insensitive caches. HPL-93-41, HP Labs, June 1993. http://www.hpl.hp.com/techreports/93/HPL-93-41.html.

[9] R. Sedgewick and P. Flajolet. *An introduction to the analysis of algorithms.* Addison Wesley, 1996.

[10] A. Seznec. A case for 2-way skewed associative caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, 1993.

[11] A. Seznec. A new case for skewed-associativity. Technical Report PI-1114, IRISA, 1997.

[12] N. Topham, A. González, and J. González. The design and performance of a conflict-avoiding cache. In *Proceedings of the 30th ACM/IEEE international Symposium, on Microarchitecture*, 1997.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399